This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

Predictive UAV Base Station Deployment and Service Offloading with Distributed Edge Learning

Zhongliang Zhao[1], Lucas Pacheco[2], Hugo Santos[23], Minghui Liu[1], Antonio Di Maio[2], Denis Rosário[3], Eduardo Cerqueira[3], Torsten Braun[2], Xianbin Cao[1]
School of Electronic and Information Engineering, Beihang University, China[1]
Institute of Computer Science, University of Bern, Switzerland[2]
Institute of Technology, Federal University of Pará, Brazil[3]
Email: {zhaozl, liumh1213, xbcao}@buaa.edu.cn, {lucas.pacheco, hugo.santos, antonio.dimaio, torsten.braun}@inf.unibe.ch, {denis, cerqueira}@ufpa.br

***Abstract*—In modern networks, edge computing will be responsible for processing and learning from the critical network- and user-generated data, such as wireless link usage, mobility information, application requests, and many others. The presence of Artificial Intelligence-based (AI) applications at the edge of the network will enable the network to predict necessary user behavior and its impact on network infrastructure, such as base station overloading. One of the main strategies for offloading users and base stations is to deploy UAV base stations, or flying base stations, which can dynamically provide service and connectivity. In this article, we introduce a framework for distributed learning over Multi-access Edge Computing (MEC), which manages data applications in a fully distributed setting across edge servers, thus reducing the cost of collecting user information in a centralized server. We couple the proposed distributed learning with a novel similarity metric for user trajectories, which can aggregate neural network models with similar costs as other model aggregation techniques. However, the aggregation technique can achieve much higher accuracy. Furthermore, we apply the proposed distributed learning scheme to manage and deploy flying base stations to areas that experience high demand or poor user connectivity, thus optimizing connectivity in terms of user satisfaction, delay, and network throughput.**

***Index Terms*—Distributed machine learning, trajectory prediction, unmanned aerial vehicle, flying base station deployment, mobility management.**

I. Introduction

With the emergence of Unmanned Aerial Vehicles (UAVs), flying base stations can be regarded as a promising solution to provide extensive coverage services to assist mobile users with limited or no wireless connectivity [1]. In this way, a UAV as flying Base Station (UAVBS) enables to offload data traffic from a set of mobile users connected to a congested base station or user located in the void area, improving the connectivity and Quality of Service (QoS). In future mobile networks, mobile users could be connected to a given base station or UAVBS in a transparent way based on a mobility management decision [2]. For instance, there are ongoing standardization activities in 3GPP [3] for providing enhanced wireless connectivity to personal and commercial UAVBSs via mobile networks [1]. Many works have been done to deploy UAVBSs to provide temporary network services [4], [5]. Hence, it is essential to understand the network scenario and user behavior to proactively deploy UAVBS to serve ground users. While UAVs will play an essential role in providing service and connectivity to users in remote areas or areas that are overloaded, they impose new mobility management challenges [6]. With the presence of UAVs as flying base stations, users' anchor points must be carefully managed to guarantee good QoS for all users in the network.

Machine Learning (ML) brings significant benefits to mobile networks to understand the network scenario and user behavior. For instance, predicting user mobility and traffic behavior is one of the main enablers in this setting, as user-generated events (e.g., mobility, downlink usage, etc.) have significant impact on the network, which may not be foreseen in the initial network design [7]–[9]. Hence, research investigating the application of ML to improve various communication systems is currently experiencing an incredible boom [10]. The unique features of wireless systems introduce new challenges when ML algorithms are applied to improve communication systems. In particular, these challenges are due to extensive network scale, geographically dispersed deployment, dynamic user mobility, exponentially growing data volume, congestion on the radio interfaces, and data privacy concerns. To cope with such issues, Distributed Machine Learning (DML) techniques, such as Federated Learning (FL), have been increasingly applied in the last years to wireless communications due to the improved computational capabilities of wireless devices. Therefore, ML-enabled wireless systems to start migration from a centralized to a fully distributed training workflow by splitting the model learning task on a central server into multiple geographically distributed servers [11]–[13]. FL has limited capabilities to converge in the presence of such non-IID (Independent Identically Distributed) data. Data from different users can vary greatly and be based on many different random distributions, which may have different representations in the trained neural network weights. Thus, in the aggregation phase of FL, it is required to group users with similar statistical features such that different learned features do not cancel each other when averaged.

MEC is a vital component towards future mobile networks

Zhongliang Zhao and Lucas Pacheco contributed equally to this work. Corresponding author: Zhongliang Zhao (zhaozl@buaa.edu.cn).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

to assist the deployment of UAVBSs [14]. For instance, 5G is already considering moving parts of the service-specific processing from the central cloud to edge nodes, physically close to the end users [15]. With edge computing, a set of computing infrastructures could be deployed close to end devices to provide a viable way to meet the high computation with low-latency requirements of ML on edge devices. It also provides additional benefits in terms of privacy, bandwidth efficiency, and computation scalability. An edge server could be co-located with a cellular base station, an IoT gateway, or a campus network. In mobile networks, user mobility prediction information, such as user trajectory prediction, can be deployed at the edge servers to improve the UAVBS deployment and mobility management optimization [16]. In this sense, edge computing provides a platform for deploying DML for wireless network management. However, how to proactively deploy UAVBSs and how to efficiently perform mobility management by incorporating ML on distributed edge computing infrastructure to decrease the data retrieval cost remain still open research questions.

This article introduces the PRedictive- and groUp mobility-based UAVBS Deployment in Edge-enabled mobile NeTworks (PRUDENT). It takes advantage of the computing capabilities of an edge-enabled network to introduce a distributed learning model for predicting network usage and user mobility with higher accuracy compared to traditional DML models. Furthermore, we introduce a location-aware scheduling mechanism for training the DML models in the presence of distributed data storage. In this sense, PRUDENT uses the trained ML models to optimize coverage and service provisioning by proactively deploying UAVBSs and by performing mobility management decisions.

All contributions proposed in this article constitute enablers for the maximization of the throughput for mobile users in UAV-enabled networks, as shown in more details in Sections IV and V. We consider the distributed nature of the scenario in order to propose a user trajectory similarity metric, as well as the allocation mechanism for the metric computation in a distributed edge computing scenario. Furthermore, we take advantage of the trajectory similarity information to make predictions and UAV deployment as well as manage user connectivity in the mobile scenario. The main contributions of this article can be summarized as follows:

- We propose a distributed learning framework, where edge servers act as local data owners to collect connection data between mobile users and edge servers. The framework comprises a distributed scheduling solution to decrease data retrieval costs over a distributed data storage.
- We present a proactive mobility management solution to provide user content request offloading and service provision to neighboring cells based on distributed predictions of user's future trajectories and content requests.
- We define a proactive UAVBS deployment strategy to serve mobile users in congested or void areas. With a distributed user clustering approach, we group users that are poorly served and define the minimum cluster size to be served to minimize deployment costs.
- We conduct extensive experiments, and simulation results show that our framework could consistently optimize throughput and the network's service level over different scenarios. Furthermore, we improve the distributed machine learning model's accuracy at the edge servers via a novel similarity measure.

The rest of this article is organized as follows. Section II reviews related works. Section III describes the system model that is considered in this work. Section IV details our solution to perform predictive UAVBS deployment based on distributed learning of user trajectory and traffic requests. Section V describes PRUDENT's user traffic offloading and UAVBS deployment scheme for QoS optimization. Section VI presents the simulation results. Section VII summarizes the contributions of this work.

II. Related Works

This section describes the state-of-the-art research results on wireless signal detection, mobile user mobility, and traffic flow prediction, distributed machine learning, and UAVBS deployment. We also discuss their strengths and weaknesses.

A. Machine Learning based Network Optimization

Different ML-based approaches have been published to optimize the network management aspects, such as channel estimation and spectrum allocation. Samuel et al. [17] applied Deep Learning (DL) for massive multi-input multi-output (MIMO) detection and channel estimation by unfolding a projected gradient descent method. They apply the approach to time-invariant and time-varying channels. The DL algorithm provides lower complexity than approximate message passing and semi-definite relaxation with the same accuracy and enhanced robustness. For channel encoding and decoding, Nachmani et al. [18] applied DL to the decoding of linear block codes with short to moderate block length based on recurrent neural network architectures. For end-to-end communications, Dörner et al. [19] provided a solution to train auto encoder-based communication systems for channel estimation and construction of channel models. Challita et al. [20] developed a DL-based resource allocation framework for the coexistence of long-term evolution (LTE) networks with licensed assisted access (LTE-LAA) and WiFi in the unlicensed spectrum. With Long Short-Term Memory (LSTM), each small-cell base station can decide on its spectrum allocation autonomously by requiring only limited information on the network state.

Meanwhile, modern networking deployment strategies, such as the ones from 5G and beyond scenarios, are characterized by the dense deployment of small cells for more efficient spectrum usage and higher data rates. Such deployment strategies can significantly improve the perceived data rates and connectivity for end-users. However, they also increase network management's complexity, especially in terms of mobility management [21]. While users in indoor spaces can expect the presence of access points and small cells, the dense deployment of small cells does not happen at a fast enough pace to provide demanding services for mobile users [22]. Therefore, proactive UAVBS deployment must be carefully managed to consider user mobility and to perform

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

handovers and to offload to the UAVBSs. As we can see, empowering the edge of the network with the capability to train and execute machine learning models can greatly improve network management. However, we must take into account the geographically distributed nature of edge computing.

B. User Mobility and Traffic Flow Prediction

With mobility as an intrinsic feature of mobile networks, it becomes essential to know mobile users' future locations proactively, which can significantly improve urban traffic management, route recommendation, etc. Simultaneously, with more mobility datasets available, it becomes possible and feasible to deploy location prediction services in many network applications to enable proactive mobility management, handover optimization, content migration, and resource management. Zhao et al. [23] proposed a proactive mobility management approach based on group user trajectory prediction by combining LSTM with Reinforcement Learning (RL) to automate the model training procedure. Ding et al. [24] proposed a multi-user multi-order Markov model and a multi-modal user mobility pattern prediction approach. These works consider that their social relationships influence people's mobility patterns in a practical trajectory system to make a context-based user mobility prediction. However, such works rely on a large amount of user data, which might not be immediately available for training. Feng et al. [25] proposed a short-term traffic flow prediction algorithm based on an adaptive multi-kernel support vector machine with spatio-temporal correlation. Wang et al. [26] proposed a deep attentive adaptation network model to transfer cross-domain spatio-temporal knowledge for urban crowd flow prediction. Oliveira et al. [27] proposed an adaptive demand forecasting model and a slice allocation algorithm based on the forecasting of the network resources demand to define slice structures in the most suitable fashion. Zhang et al. [28] used multitask DL based on fully convolutional networks to predict node flow and edge flow throughput considering spatio-temporal aspects of the network. As we can see, traffic and link usage prediction has been successfully applied in the state-of-the-art. However, correlating user mobility prediction and traffic prediction in order to find where in the scenario there might be a traffic deficit in terms of offered and requested throughput is still an unexplored research direction that can significantly optimize network metrics and operation.

C. Distributed Machine Learning

Many efforts have proposed DML, such as FL and partitioned learning techniques, to allow wireless devices to acquire a global model with limited data exchange or based on partial models and datasets. Mukherjee et al. [29] introduced a distributed deep neural network-based offloading strategy to minimize the weighted sum of the delay and the energy consumption in UAV-assisted MEC networks. McMahan et al. [30] presented the Federated Averaging (FedAvg) algorithm, which trains an aggregate model and does not require uploading client data to a server. However, FedAvg suffers from slow convergence with large numbers of rounds and high communication costs per round under non-identical and independent distributed client datasets [31]. Konečnỳ et al. [32] introduced the concept of structured and sketched model updates in order to reduce the quantity of uploaded data during training significantly. Lin et al. [33] proposed deep gradient compression, which reduces the redundant gradient in distributed gradient descent [34] to reduce the communication bandwidth significantly.

In addition to compression of weights, other techniques to reduce communication during FedAvg have been proposed. Yang et al. [35] presented an enhanced FL technique by proposing an asynchronous learning strategy on the user devices and a temporally weighted aggregation of the local models on the server. Furthermore, Mills et al. [36] proposed an adaptive FedAvg algorithm composed of distributed Adam [34] optimization and compression of uploaded models. It reduces total uploaded data and rounds compared to similarly compression techniques. In power control applications, Vu et al. [37] proposed a solution for cell-free MIMO systems to enable FL frameworks. The approach enables each of the iterations of FL to take place in a long coherence period to make sure that the FL operation is stable. The joint optimization of local precision, transmit power, throughput, and users' working frequency is formulated as a mixed-timescale stochastic non-convex power control problem. In QoS provisioning applications, Habachi et al. [38] proposed novel resource allocation techniques for Power Domain Non-Orthogonal Multiple Access (PD-NOMA) to jointly allocate the channel and transmit power in PD-NOMA systems. The authors developed a FL approach to allow the collaboration between the base station and Machine Type Devices (MTD) to estimate the traffic model and enable massive allocation.

D. Unmanned Aerial Vehicle Base Station Deployment

The use of UAVBSs is expected to play an essential role in future mobile networks. Compared to traditional communications, UAV-assisted networks have several appealing advantages, such as on-demand deployment without highly constrained and expensive infrastructure (e.g., cables), high flexibility by dynamically changing their positions to provide on-demand communications to ground users, and a better chance of having line-of-sight (LOS) communication links [2]. However, deploying UAVBSs in different scenarios with different goals becomes a key challenge. Alzenad et al. [39] evaluated the 3D position of UAVBSs to maximize the number of covered users by minimizing their transmit power.

Liu et al. [40] proposed a sophisticated ML-based model for deploying UAVBSs and determine their transmission power according to user mobility predictions. Even though the proposal optimizes the total achievable data rate in the system, it also assumes that each user's minimum rate requirements are equal and fixed over time. Other works [41]–[43] have proposed reactive solutions to position a set of UAVBS according to the instantaneous spatial distribution of traffic load generated by mobile users. However, the spatial distribution of traffic load can vary considerably faster than the physical relocation speed of UAVBSs, which may introduce service

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

delays. Furthermore, in the case of overloaded or void areas, deploying UAVBSs is not enough for providing better service, as a tailored handover strategy is also necessary to meet application requirements and supply the throughput required by end-users. Qian et al. [44] applied a reactive approach, which means the service offloading operation is performed after the user movement has finished, and the offloading decision is solely based on the Channel State Information (CSI) between the mobile user and the connected base stations. Moreover, the mobility models applied include only circle-road mobility and linear-road mobility for vehicle networks, which are much more regular than the human mobility model considered in our work. Based on the presented state-of-the-art analysis, we argue that the problem of detecting and offloading users located in void areas or with poor connectivity in a mobile scenario proactively is a crucial task. Furthermore, a UAVBS is a promising solution for the offloading user in such a situation, and also a tailored mobility management strategy must be in place to offload users from the overloaded base station to the neighboring base station and UAVBSs. While many works show the usefulness of ML techniques for modern network management, as well as for traffic and mobility prediction, they fail to consider certain characteristics of edge computing scenarios, such as the significant geographical distribution of servers through the scenario and the challenge of performing DML tasks in such scenario. Moreover, the use of UAVBSs jointly with a tailored mobility management algorithm is required for traffic offloading and throughput maximization. However, user mobility and traffic prediction play a crucial role in a UAVBS scenario, as it enables proactive UAV deployment. To the best of our knowledge, none of the previous work integrated a DML solution to solve UAVBS deployment combined with an efficient mobility management problem, while PRUDENT tackles every aforementioned critical issue.

III. System Model and Problem Formulation

This section explains the proposed system, its core components, their roles and the interactions among themselves.

A. System Model

Figure 1 shows the proposed network scenario for DML, which includes on-demand UAVBS deployment and mobility management in edge-enabled networks. We can observe that each individual base station in the scenario has a coverage area associated with it. However, a given base station can only serve a fix number of users within its coverage area, thus overloading the traffic in the respective cell. The scenario considers that the load for each cell is known, and UAVs can be deployed to offload the traffic of an overloaded cell by extending the coverage of a cell under lower traffic load. Furthermore, the services consumed by the users are distributed on the edge servers with respective to the cell they are associated with, and different kinds of services can coexist in the network, such as services used for network operation (e.g., the user mobility prediction). In this manner, Figure 1 illustrates the relationships between the different contributions of this article, such as the user trajectory prediction, similarity estimation and

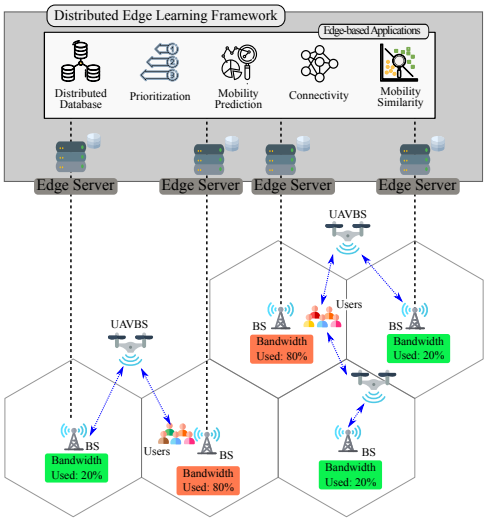


Fig. 1. Proposed distributed edge learning framework architecture, where edge servers are deployed close to the BSs to provide local data analysis, and UAVBS are deployed on demand to provide complement services.

clustering process, the UAVBS deployment, as well as mobile user mobility management, which happen on the edge servers associated with the ground base stations.

We consider a scenario containing a set $C = \{c_1, c_2, ..., c_c\}$ of cellular ground base stations that offer Internet connectivity to mobile users. In such an area, we assume the presence of a set of mobile users $U = \{u_1, u_2, ..., u_u\}$, each one with a unique ID u. We define the geographical position of a user u at time moment t as indicated by point $p_u(t)$. A given mobile user u moves across the scenario and can request a service (e.g., Virtual/Augmented Reality services, multimedia content retrieval, web browsing, etc.) from a remote service provider. Each service has different characteristics in terms of desired application's QoS, such as maximum delay, minimum throughput, average packet loss. In this work, we model a service's QoS requirements as the minimum throughput required by the application to provide the minimum service quality.

The Internet connectivity in the scenario is provided to the mobile users via the base stations C, composing a K-tier cellular network, where each tier models the cell of a particular access network, such as macrocells, small cells, or picocells. Each base station has an ID c, located at a fixed position within the scenario. *We also assume a core network with high capacity fibers connected to avoid congestion on the backhaul links, supporting all users' requested traffic without congestion.* In addition to ground base stations, we assume that our scenario also contains a set of UAVBSs V, which can be deployed on-demand to arbitrary locations in the scenario to offload the downlink of nearby base stations. We must consider characteristics of UAV networks in this setting, most notably: i) the capacity of the battery included in a given UAV, which can be extended via the presence of charging stations, or UAVs must be replaces for service continuity; ii) the presence of LOS links between UAV and users, as well as between UAV and ground base stations; iii) the time to reach a certain destination after an UAV deployment strategy is derived by the network.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

Hence, the mobile user u could consume the service-connected to a ground base station or UAVBS, based on the mobility management algorithm's decisions.

The considered scenario includes a mobile edge computing layer composed of a set of edge computing servers E with a unique ID e, responsible for data storage, the construction, and training of ML models, as well as network management tasks. Each edge computing server e is uniquely associated with a distinct ground base station according to the standard defined in Filali et al. [45]. User requests could be processed and attended by the edge servers, offering the desired content to be retrieved. The user's request history is modeled as a sequence containing user ID, size of the payload transferred at the downlink, and timestamp of the request. Edge servers must process and store significant amounts of user data, such as users' location and request history, as well as the neural network models trained for users. We assume that a given edge server e can retrieve the data of a user u for neural network training and other processing tasks from other peer edge servers. In particular, we consider a distributed storage mechanism similar to the Hadoop Filesystem (HDFS) implementation, which periodically offloads its least-frequently accessed data to a remote centralized facility. The edge storage deployment is supported by backhaul links with bandwidth W_e, which connect every couple of edge servers. Thus, a given edge server e can communicate and exchange information with a particular edge server in the network at all times. Table I summarizes the main symbols used to explain the considered edge-enabled network scenario.

In such an edge-enabled network scenario, we assume that the ML prediction models are pre-trained and stored on the edge servers. We also assume that applications in this context are executed on a specific edge server $\{e \in E\}$ that is close to mobile users. The edge computing layer is responsible for predicting future locations and down-link usage based on the DML model for individual users to assist mobility management and UAVBSs deployment decisions. These issues are introduced in the following sections.

TABLE I
SYSTEM MODEL PARAMETERS

Symbol	Description
U	Set of mobile users
C	Set of ground BSs
E	Set of edge servers
V	Set of UAVBSs
$p_u(t)$	Position of user i at time t
$r_u(t)$	Rate of user i at time t
A_u	Machine learning architecture for user m
T	User data update periodicity
M_u	Mobility history for user u
S_u	Service request history for user u
R_u	Signal-to-Noise Ratio (SINR) history for user u
R_c	Square tessellation cell which defines the spatial granularity of throughput estimation
C_r	Cost to fetch data from one user
Γ	Spectral efficiency of a base station in a given location
Θ	Throughput available from a base station in a given location

B. Problem Formulation

Let ψ denote the *service level*, which is defined as the fraction of users who receive the minimum throughput required by their applications. The definition of the minimum QoS threshold varies according to the service requirements. For instance, service level indicates the minimum throughput or the maximum latency to provide the minimum QoS to the user. The formulation of the service level metric can be found in the target function in Equation (1), where H refers to the step function, ξ represents the received data rate at the user-device application layer, ϵ represents the necessary data rate for the user's applications, and $\lambda \in [0, 1]$ is a lenience factor as to the fraction of the datarate that must be received to count a user as served (i.e., the minimum required for the application functioning). We assume that an user is not "served", as soon as the network cannot provide a QoS higher than the minimum threshold. We formulate the problem addressed by PRUDENT as shown in Equation 1, and we can see that the maximization of network throughput is the main optimization problem to be solved by PRUDENT.

$$\begin{aligned} \max_{\psi \in [0,1]} \quad & \frac{1}{U} \sum_{u \in U} H(\xi_u - \epsilon_u \times \lambda) \\ \text{s.t.} \quad & \sum_{u \in U'} \xi_u < \zeta \forall c \in C \end{aligned} \tag{1}$$

We further model the parameters of the system in terms of the throughput achievable by users connected to the network, the tessellation of the coverage areas of the scenario, and the strategies to improve ψ in Section V. In the following, we provide the details of the proposed user mobility and network usage prediction with the distributed machine learning architecture.

IV. Network Usage and User Mobility Prediction with Distributed Machine Learning

This section introduces PRUDENT's DML scheme, which consists of a distributed user data storage at the edge servers and the scheduling of neural network training jobs with this architecture. In the DML training step, user data is considered to train ML models to predict users' location and service requests at the edge server. In a distributed edge computing scenario, data locality can impact the performance of the learning process. In this context, most of the user data generated at the edge of the network tend not to leave the edge to be stored in a cloud environment. Such data can be stored at edge servers distributed through the scenario, and there is a communication cost associated with moving such data to the cloud or even to other edge servers. The necessity for distributed learning arises from the limitations of individual edge computing servers to process the entire dataset. Distributed user similarity search finds similar statistical features, which are learned similarly by given neural network architecture. In the prediction step, the user mobility and request prediction models are used to forecast overloaded cells and areas in the scenario.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

A. DML Models Training

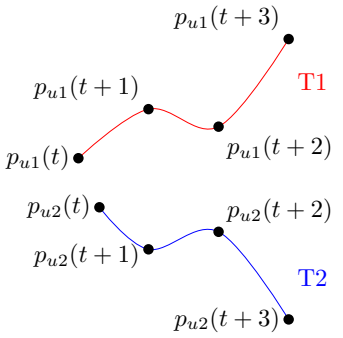
In this step, the PRUDENT execution requires pre-existing user data at the edge layer to train ML models in a distributed fashion to predict users' future location and downlink usage at the edge servers. In this context, a user is modeled as a quadruple composed of: i) its user ID u, ii) its *mobility* history M_u, iii) its *downlink usage* history S_u, and iv) history of the SINR of the signals R_u received by the user's serving and neighboring base stations. We assume that users periodically report their information to the network edge servers. Afterwards, PRUDENT applies a clustering algorithm to group users with similar data features at the edge server. For instance, neural network models are aggregated over ID users, reducing the number of models stored in the network. This helps to maintain higher accuracy than in traditional distributed learning techniques, which may train a single model for non-ID users. For data storage, we consider a Distributed Multi-Level Storage (DMLS) deployed at the edge of the network in which edge servers function as remote storage servers.

B. Distributed User Similarity Search

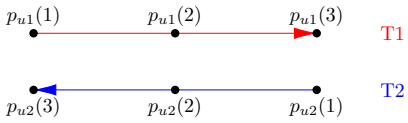
Given the limited storage capabilities of the edge servers, we introduce a distributed similarity search scheme by analyzing users' downlink usage and mobility to group machine learning models. Model grouping is required because the system may need to store several thousands of parameters for each machine learning model, which could introduce a high storage cost. Thus, it is advantageous to group models if the system can maintain low error rates. Traditionally, model aggregation is made by averaging the weights of all trained models into a single model. This approach has the disadvantage of grouping together machine learning models with potentially very different and contrasting statistical features, which may negatively impact error rates and accuracy. In this article, we propose intermediate clustering processing to decrease the number of trained parameters to be stored by the network while maintaining high accuracy levels.

The first operation performed by PRUDENT on user data is a distributed similarity search: user pairs are scored according to a similarity metric. This operation is beneficial because it groups users with similar mobility patterns to aggregate neural network models with similar statistical features. PRUDENT performs the similarity search for both the user mobility data and the user link usage information. In this manner, two users with similar overall trajectories in their data sequences will have a higher similarity value concerning mobility, and users with similar service request patterns will have a higher similarity value concerning their requests.

We consider the Longest Common Sub-sequence (LCSS) [46] for trajectory similarity, which is a reliable similarity metric to measure the similarities of trajectories. We denote the trajectories of two mobile users u_1 and u_2 as $T_{u1} = (p_{u1}(1), \ldots, p_{u1}(n)) \in \mathbb{R}^{2n}$ and $T_{u2} = (p_{u2}(1), \ldots, p_{u2}(m)) \in \mathbb{R}^{2m}$ with $p_{u1}(i), p_{u2}(j) \in \mathbb{R}^2, \forall i, j$. We define the operator $h : \mathbb{R}^{2n} \rightarrow \mathbb{R}^{2(n-1)}$, which returns a trajectory made of the first $n-1$ points of a trajectory made of n points. Equation 2 defines $L(T_{u1}, T_{u2}) \in \{1, \ldots, \min(n, m)\}$ as the LCSS between two trajectories T_{u1} and T_{u2}. The constants γ and δ are the matching threshold and the time shift, respectively, and can be adjusted offline according to the scenario's characteristics.



(a) Two trajectories with different shapes



(b) Two trajectories in the opposite directions

Fig. 2. Limitations of the LCSS algorithm.

$$L(T_{u1}, T_{u2}) = \begin{cases} 0, \quad \text{if } \dim(T_{u1}) = 0 \text{ or } \dim(T_{u2}) = 0 \\ 1 + L(h(T_{u1}), h(T_{u2})), \\ \text{if } \|p_{u1}(n) - p_{u2}(m)\| < \gamma \text{ and } |n - m| \leq \delta \\ \max \begin{cases} L(h(T_{u1}), T_{u2}) \\ L(T_{u1}, h(T_{u2})) \end{cases}, \text{otherwise} \end{cases} \tag{2}$$

Equation 2 is the recursive formulation of the LCSS algorithm, and its rationale is detailed hereafter. The second expression of Equation 2 means that if the time distance between the two last data points of the two trajectories is less than a threshold δ and they are located closer to each other than a threshold distance γ, the LCSS associated to those trajectories is recursively computed by adding 1 to the LCSS score associated to the two trajectories. The third expression of Equation 2 states that if the second expression's conditions are not satisfied, i.e., the two last data points are further apart than the threshold γ or the trajectories' length difference is more than δ, the LCSS score associated to the two considered trajectories is the largest LCSS between the same two trajectories, computed after removing the last data point from either of them. Finally, the first expression of Equation 2 states that when the algorithm calculates the LCSS between two trajectories, of which at least one contains no data points, it will return an LCSS value of zero, and this will terminate the recursion.

Equation 3 defines the similarity between two trajectories T_{u1} and T_{u2} based on LCSS.

$$\sigma_L(T_{u1}, T_{u2}) = \frac{L(T_{u1}, T_{u2})}{\min(n, m)} \tag{3}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

The LCSS algorithm does not consider the shape of trajectory segments, as it uses only the distances between points on the two different trajectories, as location data can be noisy, and traditional approaches do not consider the direction of movement. For instance, two trajectories $T_{u1} = (p_{u1}(1), p_{u1}(2), \ldots, p_{u1}(n))$ and $T_{u2} = (p_{u2}(1), p_{u2}(2), \ldots, p_{u2}(m))$, shown in Figure 2(a), are significantly different, especially after the intersection point P, but their similarity value will be high according to the LCSS algorithm. Figure 2(b) shows an extreme example that satisfies the condition $\|p_{u1}(1) - p_{u2}(1)\| < \gamma$. The two trajectories are nevertheless considered highly similar and grouped with the conventional LCSS algorithm, but they represent opposite directions. Therefore, calculating trajectory similarity using only the distance between data points on trajectories will not consider all the relevant information of user data.

To solve this issue, we hereafter propose a novel similarity measure between user mobility trajectories. In order to improve the accuracy of similarity measurements, we measure the similarities between pairs of segments instead of individual data points distance on the trajectory. The i-th segment on a trajectory is defined as a directed vector $\mathbf{v}(i) \in \mathbb{R}^4$ in a two-dimensional space $\mathbb{R}^2$ and can be represented by a couple of points $(p(i), p(i+1))$ on the $\mathbb{R}^2$ plane. Specifically, $p(i) \in \mathbb{R}^2$ is the point where the segment starts, and $p(i+1) \in \mathbb{R}^2$ is the point where the segments end. We define the distance metric $d : \mathbb{R}^4 \times \mathbb{R}^4 \rightarrow \mathbb{R}$ between segment $\mathbf{v}_{u1}(i)$ and segment $\mathbf{v}_{u2}(j)$ as the sum of Euclidean distances between the two pairs of endpoints, as shown in Figure 3 and defined in Equation 4. This definition of distance between two segments satisfies the properties of positivity, symmetry, and triangular inequality.

$$d(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) = ||p_{u1}(i) - p_{u2}(j)|| + ||p_{u1}(i+1) - p_{u2}(j+1)|| \tag{4}$$

The distance similarity σ_d is defined in Equation 5, where α is a random variable between [0, 0.1] to ensure that the denominator is not zero when two segments overlap.

$$\sigma_d(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) = \frac{1}{d(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) + \alpha} \tag{5}$$

The angle similarity σ_θ between two segments is based on cosine similarity and is defined in Equation 6.

$$\sigma_\theta(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) = \begin{cases} 0, & \mathbf{v}_{u1}(i) \cdot \mathbf{v}_{u2}(j) \leq 0 \\ \frac{\mathbf{v}_{u1}(i) \cdot \mathbf{v}_{u2}(j)}{|\mathbf{v}_{u1}(i)||\mathbf{v}_{u2}(j)|}, & \text{otherwise} \end{cases} \tag{6}$$

where $\frac{\mathbf{v}_{u1}(i) \cdot \mathbf{v}_{u2}(j)}{|\mathbf{v}_{u1}(i)||\mathbf{v}_{u2}(j)|}$ is the cosine of the angle between $\mathbf{v}_{u1}(i)$ and $\mathbf{v}_{u2}(j)$. If the directions of the two segments are widely different, it makes no sense to compute the distance of the segments. Thus, $\sigma_\theta(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j))$ is set to 0 for θ greater than $\pi/2$. Finally, we define the segment similarity metric σ as the product of metrics σ_θ and σ_d as in Equation 7.

$$\sigma(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) = \sigma_\theta\left(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)\right) \cdot \sigma_d\left(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)\right) \tag{7}$$

Based on the proposed segment similarity, PRUDENT classifies each pair of users in order to cluster them. Specifically,

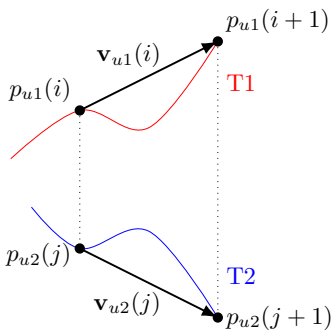


Fig. 3. Similarity between trajectory segments

Algorithm 1: Trajectory Similarity Measurement

1 **for** *each user pair* P_t **do**
2 $V_{u1} = (\mathbf{v}_{u1}(1), \ldots, \mathbf{v}_{u1}(n-1))$;
3 $V_{u2} = (\mathbf{v}_{u2}(1), \ldots, \mathbf{v}_{u2}(m-1))$;
4 V_{u1},V_{u2}: Sets of directed vectors for T_{ut1} and T_{ut2};
5 **for** $i = 1$ *to* $n-1$ **do**
6 **for** $j = 1$ *to* $m-1$ **do**
7 **if** $i = 1$ *or* $j = 1$ **then**
8 $D(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) = 0$
9 **else**
10 **if** $\sigma(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) > \tau$ *and* $|n-m| \leq \delta$ **then**
11 $D(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) = 1 + D(\mathbf{v}_{u1}(i-1), \mathbf{v}_{u2}(j-1))$
12 **else**
13 $D(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j)) = \max\{D(\mathbf{v}_{u1}(i-1), \mathbf{v}_{u2}(j)),$
14 $D(\mathbf{v}_{u1}(i), \mathbf{v}_{u2}(j-1))\}$
15 **return** $\Phi_t = \frac{D(\mathbf{v}_{u1}(n-1), \mathbf{v}_{u2}(m-1))}{\min(n-1, m-1)}$

considering a pair of users as a 2-tuple composed of users u_1 and u_2, we must apply the proposed similarity metric over each pair of users to score how similar the users are.

In our scenario, we consider a system contains a set of U users, and there are $N_p = \frac{|U|^2 - |U|}{2}$ distinct pairs of users, where such calculation can be performed in parallel to take advantage of the distributed nature of edge computing. Let us define, for each pair of users $P_t | \forall t \in \{1, 2, \ldots, N_p\}$, a job consisting of the calculation of trajectory similarity in Algorithm 1, where u_{t1} and u_{t2} are the users within P_t. In the proposed algorithm, we define $D(\mathbf{v}_{u1}, \mathbf{v}_{u2})$ as the LCSS between $\mathbf{v}_{u1}$ and $\mathbf{v}_{u2}$ in P_t. The similarity value between a pair of users is represented by $\Phi_t\,[0, 1]$. The constant τ, shown in line 10, is the segment similarity threshold. A large value of τ will lead to better precision but smaller recall results. For careful consideration, τ can be selected by F1-measure. Note that for trajectories of lengths m and n, the complexity of the comparison operation will be $O(n \times m)$. In terms of convergence, the LCSS algorithm is based on the construction of a matrix which is filled with the largest found

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

sequence at each step of the algorithm. Thus, the algorithm converges to the largest sequence in $m \times n$ steps, as it is always necessary to consider all elements of each sequence. The cost of execution may vary based on the policy used to consider that two sequences, defining an optimality model for the algorithm may be complex as the yielded results differ in nature as such policy is changed.

Algorithm 2: Similarity Search Scheduling

```
1  for each user pair P_t do
2      S_u1 = Set of servers which contain the data for u_t1;
3      S_u2 = Set of servers which contain the data for u_t2;
4      if S_u1 ∩ S_u2 ≠ ∅ then
5          for Each server in S_u1 ∩ S_u2 do
6              if Server has available computing resources
                 then
7                  Allocate job P_t to server;
8                  Break;
9      for Each server in S_u1 ∩ S_u2 do
10         if Server has available computing resources
             then
11             Retrieve data of missing user;
12             Allocate job P_t to server;
13             Break;
14     if Pair P_t not allocated then
15         Query edge server with available computing
             resources;
16         Retrieve data if user u_t1;
17         Retrieve data if user u_t2;
18         Allocate job P_t to server;
```

We define a scheduling policy for the calculation of all N_p similarity values in the system. For such, we consider the locality of the data. As previously stated, each data piece is stored at the edge of the network and has ρ copies of itself. The data of a specific user can be found at a given server with probability $\frac{\rho(1-p_d)}{C}$, where p_d represents the probability that the user data is no longer available on the specific edge server.

The cost for a given edge server to retrieve certain user data u is denoted by C_r and is considered to be uniform across the network (i.e., the cost is the same for any server to retrieve any user who is not stored locally at the server). We propose Algorithm 2 to minimize the costs for retrieving a user not stored locally at the server in which the computation will occur. In the proposed scheduling strategy, we prioritize edge servers that are storing the data of specific users to perform the learning process over such data. In this, we do not spend resources in retrieving user data from a remote server, as shown in Algorithm 2, lines 4 to 8. However, if no servers have the data for both users locally, we consider all the servers that have the data for one of the users. In that case, if any of the servers with data from one of the users have the available resources, we retrieve the data of the user not locally stored and perform the computation, with cost C_r to retrieve the user data, as shown between lines 9 and 13 of the algorithm. As a fallback case, if none of the servers that have user data locally stored have the necessary resources, we allocate the job to the first server found with the necessary resources, with cost $2C_r$ to retrieve the user data, as shown between lines 14 and 18. Note that the complexity of the job allocation is proportional to the product between N_p and the number of edge servers. In the case of Algorithm 2, we find that the execution complexity is proportional to the product between the number of users and the number of edge servers available for computation in the network. We consider such to not have a significant impact in the algorithm execution, as the scheduling and similarity estimation processes can happen offline or at the initialization of the system, and iterate over the existing user data at the moment.

After the similarity values for all pairs of users have been calculated, we apply a clustering algorithm to group users with high similarity indexes among themselves. By clustering users, we obtain a number of subsets composed of similarly distributed user data. For instance, users with similar mobility patterns or with similar service request patterns are more likely to be part of the same cluster. In this sense, users in the same cluster have a stronger similarity among themselves. While many different clustering algorithms can be applied for this problem, we apply Spectral Clustering (SC) in the PRUDENT functioning, as it can be easily used with the pre-computed affinity matrix obtained, as well as having a low complexity under a predefined number of clusters.

C. Distributed Mobility and Traffic Flow Prediction

The user clusters found in the distributed user similarity search step are composed of users with similar statistical features, which are learned in a similar manner by a given neural network. With the proposed framework, we consider two categories of data on which learning is applied: *(i)* user mobility data, and *(ii)* user service requests. While other categories of data can be used in the proposed ML framework, the complexity of integrating another data sequence may have diminishing returns compared to the ones chosen.

Similar to the distributed user similarity search operation, we take advantage of the distributed nature of the computing power and storage in the network by creating a series of neural network training jobs to be distributed over the participating edge servers. We consider the edge servers in the network to be homogeneous in terms of computing power and connectivity. In that sense, all servers can participate in the training process.

We assume an existing neural network architecture search scheme to be present in the network, which will decide on the best neural network architecture for the system based on the user data available. For each of the trainable data categories, a neural network architecture is chosen by a Neural Architecture Search (NAS) in place. In this sense, we consider the two distinct architectures: i) $Arch_{mobility}$ takes as input a sequence of user coordinates and outputs a pair of coordinates predicted for the next time step; ii) $Arch_{requests}$ takes as input a sequence of past service requests and outputs the request on the next time step. All hyper-parameters of the architectures are chosen by the NAS system, and the base architectures are stored over the network.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

With the pre-existing base architectures, we must define a neural network training scheduler. The scheduler is a generalization of the one presented in Algorithm 2 for the case of retrieving data from a single user instead of a pair of users. The scheduler takes into account the locality of the data in the network by attempting to train a given user u at the same edge server e in which the data from u is stored. Thus, for each user u, we iterate over the ρ servers, that possess the user's data if such a date has not been sent to remote storage. If any of such servers have the necessary computing resources for the training, the framework allocated the training job to the server in question. However, if the user data has been sent to remote storage, or if the servers that possess the user data do not have the necessary resources, the framework will allocate the training job for the first server with available computing resources available.

In this sense, user models that have been individually trained are aggregated into more general neural network models by performing a parameter-by-parameter averaging so that the aggregated model has parameters from all its participating users. Aggregated user models not only result in lower transmission and storage costs over the network but can also be reused (e.g., a new user joining the network can inherit a pre-existing model, and transfer learning strategies can be applied to speed up the training process). Such clustered neural network models are stored in the edge computing DML and are retrieved when the system must perform a prediction for a given user. In our view, PRUDENT retrieves the clustered neural network and estimates the future user coordinates and downlink usage based on historical user data.

V. Proactive UAVBS deployment with Edge-enabled DML Services

In this section, we introduce the PRUDENT scheme with the aim to maximize the network service level. This is achieved by detecting in which areas of the network it is important to offload the user-to-infrastructure traffic to UAVBSs. We start by describing how PRUDENT estimates the available throughput in the network as a function of user location to detect overloaded and void areas in the ground base station deployment. Afterwards, we describe how to offload the traffic from such users to neighbor cells and by deploying UAVBSs together with mobility management.

PRUDENT takes into consideration the users' radio measurements, as well as distributed predictions of users' future trajectories and content requests provided by the prediction step, in order to serve mobile users in overloaded or void areas. PRUDENT operations depend on three main decisions: *(i)*: UAVBS deployment to decide the number and the positions of a set of UAVBSs, which are deployed to cover and offload the traffic from a set of users in a certain area; *(ii)*: mobility management to decide whether user traffic should be offloaded to a UAVBS or keep connected to a ground base station; and *(iii)*: mobility management to connect the UAVBS to a ground base station as their backhaul link.

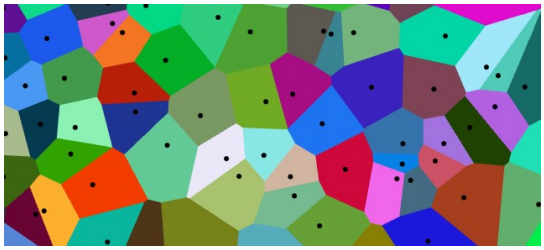


Fig. 4. Genereric voronoi tessellation for the considered scenario.

A. Detection of Overloaded or Void Areas

The first step in the optimization process is to proactively predict which area of the scenario might be overloaded by user requests. We model the scenario as an Euclidean space on $\mathbb{R}^2$, and we partition it into a number of *Voronoi cells* containing exactly one fixed ground base station c each. Equation 8 defines the generic Voronoi cell $V(c)$ associated to the base station c.

$$V(c) = \{x \in \mathbb{R}^2 | \forall c' \in C : d(x,c) \leq d(x,c')\} \qquad (8)$$

Each cell $V(c)$ can be interpreted as the set of positions in the scenario (or points in the $\mathbb{R}^2$ space) closer to c than any other base station. Due to its mathematical properties, the union of all Voronoi cells completely covers the scenario without overlapping or, equivalently, $\forall x \in \mathbb{R}^2 : \exists! c \in C | x \in V(c)$ and $\bigcup_{c \in C} V(c) = \mathbb{R}^2$. As in a real deployment, each user u is uniquely associated with a single cell (i.e., to a single ground base station c). As an example, Figure 4 shows the *Voronoi tessellation* (i.e., the process of dividing the scenario in voronoi cells) for a generic scenario containing C base stations. We can see that the cells' borders lay on the orthogonal bisecting lines of each segment that connects the base station c to its other neighboring base stations. Every user located in each colored region can communicate only with the single base station associated with that cell.

At the beginning of the algorithm execution, we assume that the considered scenario is an isotropic space with monotonic decay of transmission power from the signal source. Under this assumption, every user u is therefore served by the base station c associated with the Voronoi cell in which the user is located. Under PRUDENT, users do not necessarily have to be attached to the base station with the highest SINR values.

Let us assume that the scenario is partitioned into identically-large square cells R_c, and the set of all R_c is ρ. We consider each tessellation cell R_c to be small enough that the received SINR in a given cell does not vary significantly. In this sense, we consider the SINR $S(R_c)$ in each of the square cells R_c to be constant at any position within the cell. Therefore, it is also reasonable to assume that the throughput offered by the base stations c to the user u is also constant at any position within the base station area R_c. In this case, smaller cells will lead to better performance of the proposed solution at the cost of an increased computational load on the edge servers. The cell size R_c should be decided according to the available computing power of the deployed edge servers. Equation 9 presents the definition of spectral efficiency $\Gamma(R_c)$ of the network within the cell area R_c. It represents how

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

efficiently the physical and MAC layers of the user devices use the communication channel.

For a given base station detectable in R_c, we define the link bandwidth W as the link bandwidth offered by the base station, such that bandwidth corresponds to the Shannon capacity (i.e., the theoretical limit of the data rate in the channel) of the system and depends on the frequency used by the base station, the transmission mode used (i.e., SISO or MIMO), the number of antennas used, and other parameters. We consider W to be already known at system startup. Thus, the available throughput at a given location is proportional to the link bandwidth W and to the Spectral Efficiency $\Gamma(R_c)$ at the location. We also define the normalized average signaling overhead $U_c \in [0, 1]$ as the average ratio of the bandwidth used by users to transmit network control information. We finally define the throughput $\Theta(Rc)$ offered to an user u located in the cell R_c as in Equation 10, which is derived from the offered throughput definition proposed in Arshad et al. [47] and the work by Demarchou et al. [48]. We can observe that the throughput of a given point of R_c is proportional to the spectral bandwidth in R_c, and inversely proportional to the cost of signaling U_c, given by the fraction of the bandwidth used to transmit control messages. We assume that the average user speed, the average Voronoi cell density, and the handover-induced delay are such that the handover cost term is zero.

$$\Gamma(R_c) = \int_0^{\infty} \mathbb{P}\left(\log_2(1 + S(R_c)) > z\right) dz \tag{9}$$

$$\Theta(Rc) = W \cdot \Gamma(Rc) \cdot (1 - U_c) \tag{10}$$

In summary, we compute the throughput for each user u for all base stations within the user radio range. For each base station c, we divide its coverage into multiple identically large square cells R_c. The square cells R_c value must provide a good granularity while being computationally efficient for base stations with high transmission power. The offered throughput $\Theta(Rc)$ in a given square cell R_c is assumed to be constant at any point, reducing the computation required by PRUDENT. In this way, a given user u located with any point of square cell R_c can benefit from a throughput $\Theta(Rc)$, as shown in Figure 5. Finally, the user u must be connected to the base station c that offers the highest throughput $\Theta(Rc)$ in such location $p_u(t)$. The offered throughput map can be computed offline by the network infrastructure, as the ground base stations' positions and characteristics do not change over time.

PRUDENT detects overloaded areas in the scenario by correlating the user's current and future link usage to the actual throughput in the user's location. For each user in the network, we assess the user location and the R_c cell in which the user is. Afterwards, we check if the available throughput from any detectable base station in R_c is sufficient to provide the necessary throughput for the user. Furthermore, PRUDENT tries to detect other bottlenecks such as the backhaul of the available Base Stations (BSs) in the user location. Based on such calculations, we detect void and overloaded areas in the base station deployment. In this sense, we assume that an user placed in such areas could not be "served", as soon as the network cannot provide the required QoS for a given service. Hence, it is important to offload user traffic in such areas to neighbor cells and to UAVBS, which is achieved by UAVBSs deployment and mobility management operations performed by PRUDENT.

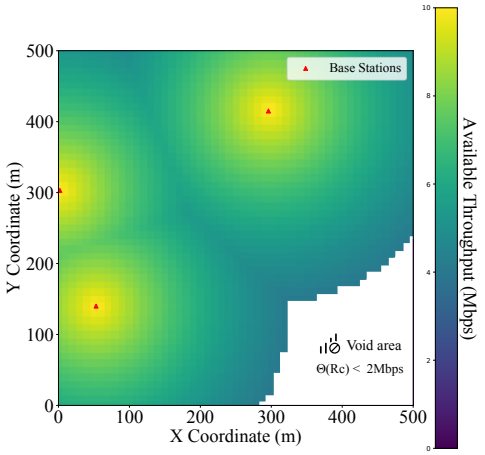


Fig. 5. Void areas under a $2\,\mathrm{Mbit/s}$ user service request

B. Mobility Management

Algorithm 3 details the mobility management and UAV positioning scheme, which are responsible for estimating the service level of the network, detecting unserved users, and clustering users for better UAV deployments. From lines 1 to 11 of Algorithm 3, the network must estimate the service level and detect unserved users. This is done through the throughput estimation as given in Equation (10). Following this step, in lines from 12 to 18, the network clusters unserved users for the allocation on UAVs and calculates the centroid of each cluster proportionally to the individual requirements of users, as given by Equation (11).

$$P_{centroid} = \frac{1}{k} \sum_{i=0}^{k} \psi_i x_i \tag{11}$$

As described, the mobility management step of the algorithm is tailored towards the offloading of unserved users with the use of flying base stations. Thus, the offloading decision responsible for performing handover for unserved users to a UAVBS can coexist in the network with a standard handover policy applied to users served normally within their requirements. While the standard handover policy in the network can be arbitrary, we consider that the user offloading proposed in PRUDENT can override the default mobility management. Performance metrics for the proposed mobility management are given in terms of throughput maximization achieved by the user offloading, as well as by the service level of the network, while other metrics may depend more heavily on the default handover policy. PRUDENT offloads users from overloaded base stations to neighboring non-overloaded base stations. This is required due to the limited number of UAVBSs available for positioning, the configuration of a directional backhaul link, and others. PRUDENT performs a user-by-user handover decision for users located in the overloaded and void

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

Algorithm 3: Mobility Management

```
Unserved Users ← ∅;
for Each user in the network do
    Perform forward propagation on prediction models;
    for Each predicted location and throughput of user do
        if BS responsible for Voronoi cell cannot serve predicted user requirement then
            for Each BS detectable do
                if Considered BS can provide the service throughput requirements then
                    Perform handover;
                    Continue;
                else
                    Unserved Users → Unserved Users ∪ {user};
Cluster unserved users based according to geographical positions;
for Each user cluster do
    Calculate Centroid of Cluster;
    Allocate closest unused UAV to centroid;
    Flag allocated UAV as in use;
    Connect UAV to a directional link of closest non-overloaded cell;
    Perform handover of unserved users to allocated UAV;
```

areas, maximizing the service level in the scenario, as shown in Algorithm 3. In the mobility management operations, we create an empty set of all users not being served with their minimum requirements. Then, for each user, we perform a forward propagation on their trained ML models in order to estimate the future positions and network usage from the user services periodically.

At each prediction, we evaluate the expected base station for the user to be connected, according to a traditional signal-based handover algorithm. This corresponds to the base station that yields the Voronoi cell in which the user is at the moment. PRUDENT will only perform a signal-based handover for a given user u, as soon as the user's expected base station is not able to provide the necessary throughput for the user's services, according to the previously calculated throughput map (Section V-A). Note that PRUDENT now knows that at a given moment, such user will be consuming a known base station's bandwidth, so we subtract this amount from the theoretical capacity of the base station can provide, considering its backhaul link and access layer. Hence, we model the base station capacity corresponding to the minimum between the access layer capacity of the base station and the backhaul link capacity for the base station being considered.

C. UAV Deployment

Even after performing handovers from users of overloaded base stations to neighboring cells, there are users who cannot be properly served by the existing infrastructure. User offloading via UAVBSs is applicable in the case of the sudden increase in traffic volume, which cannot be quickly, or economically viable to fix via ground base station deployment. In such cases, PRUDENT offloads user traffic from the overloaded base stations to UAVBSs. We define $U_{unserved}$ as the set of under-served users that cannot utilize the link capacity necessary for their applications, which cannot also be offloaded to another base station. This can be due to insufficient link bandwidth or due to a congested backhaul link at the base station. In the first step of the UAVBS positioning decision, PRUDENT considers the geographical position of the under-served users $U_{under-served}$ in the network. In the case that users are not properly served due to one or more overloaded cells, under-served users will be grouped close to each other.

While PRUDENT is able to predict when a user is going to be under-served within a time window t, it is necessary to also evaluate which UAVBS can be allocated to the user or a group of users. UAVBSs require a directional link to a ground base station to work as backhaul since they can effectively work as a relay to a base station with lower load, increasing the available throughput at overloaded locations.

Under-served users in the network are clustered in order to maximize the number of users served by a UAVBS. PRUDENT considers the K-means clustering algorithm to group together users not receiving their minimum application requirements, in which the number of clusters formed is initially set as the number of UAVBSs available in the network. In this sense, each user in the $U_{unserved}$ set is part of V clusters formed by PRUDENT.

After pruning small clusters, we have $V' \leq V$ user clusters. For each cluster, PRUDENT computes the cluster's weighted centroid as given in Equation (11), where k is the number of users belonging to the cluster, ψ_i is the throughput requested by the i-th user, and x_i is the i-th user position. Then, UAVBSs are deployed at each centroid location to serve the under-served users in the cluster so that the UAVBS is closer to the users requesting greater bandwidth. This way, more demanding users will have a better link quality with the UAVBS, and therefore they will be able to achieve a greater throughput. For each cluster centroid, PRUDENT queries the closest UAVBS available and requests the UAVBS to move to the centroid being considered. When a UAVBS is allocated to a user cluster, it is flagged as being used for the duration of the cluster and is not considered for use in subsequent clusters.

VI. Performance Evaluation

This section describes the simulations conducted over different mobility and edge management techniques to evaluate PRUDENT's performance against a set of state-of-the-art UAVBS positioning strategies.

A. Simulation Setup

We evaluated PRUDENT in a simulated network environment using realistic user mobility traces in the city of Köln (TAPASCologne scenario [49]), as well as for the deployment

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

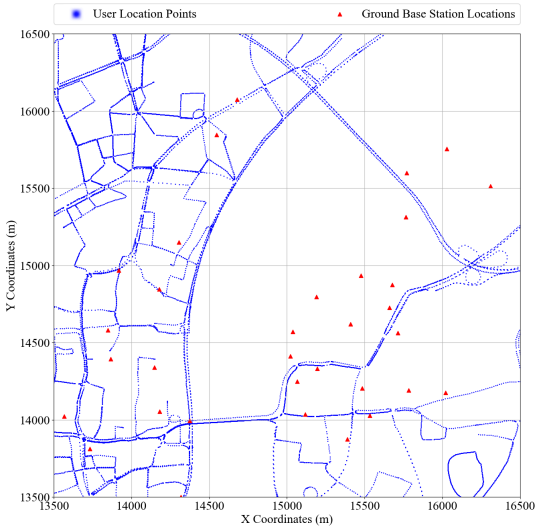


Fig. 6. The simulated Köln scenario, where blue dots represent the user positions and red dots represent the ground base stations.

of ground base stations. The Köln mobility trace consists of a large simulation of vehicular mobility, which can be integrated with network simulators as input for the node positioning and mobility. The networking and mobility simulations have been carried out using ns-3[1] and SUMO[2], respectively. We consider a partition of the Köln scenario into square cells with $2\,\mathrm{km}$ edge. The simulated user mobility and the ground base station deployment are taken from [49], which provides synthetic mobility information about several thousands of vehicles through a period of 24 hours in a $20\,\mathrm{km} \times 20\,\mathrm{km}$ area around the city of Köln. The use of SUMO as a realistic traffic simulator is required by the TAPASCologne (i.e., the Köln mobility dataset) in order to generate mobility traces for a large number of vehicles in the given scenario. This allows us to simulate a realistic network based on SUMO-generated empirical data for user mobility. From such vehicles, we randomly choose the number specified in the simulation and use their generated mobility in ns-3 as inputs to nodes positions and mobility, as well as the positions of base stations, which are also given by the dataset, constituting a highly realistic mobility scenario. However, in the present work, we restrict our scenario to a $3\,\mathrm{km} \times 3\,\mathrm{km}$ squal near the city center, as represented in Figure 6. Note that in Figure 6, the blue dots represent every location where a user is recorded in the dataset, which marks all user locations registered within the shown area.

In this context, UAVBSs are first randomly allocated through the scenario, and when necessary, a UAVBS is reallocated by the management algorithm in place. We assume that UAVBSs are charged via a distributed laser-charging system, as described by Liu et al. [40], in which directional charging stations can be mounted at ground base stations and be made available jointly with backhaul links supplied by cells.

Furthermore, downlink usage and requests made to edge computing servers are modeled according to usage from the real High Power Computing (HPC) trace showcasing computing jobs submitted to the Google Cloud infrastructure [50], which describes the jobs in terms of user, time of the allocation, memory usage, and other features. In this way, we selected individual users and assigned the jobs submitted by such users to individual users in the simulation scenario. We conducted 33 simulations with different randomly generated seeds by the simulator's default pseudo-random number generator (MRG32k3a). Results show the values with a confidence interval of 95%. Table II summarizes the main simulation parameters, which describes the key features of UAVs.

TABLE II
SIMULATION PARAMETERS

Parameters	Values
Number of UAVs	10
Number of Ground BSs	31
Number of Users	{30, 60, 90}
Mobility Model	Köln mobility [49]
BS Deployment	Köln BS deployment [49]
UAVs Speed Range	$10\,\mathrm{m/s}$ to $16\,\mathrm{m/s}$
UAVs Transmission Power	16 dBm
UAVs Height	$30\,\mathrm{m}$ to $40\,\mathrm{m}$
UAVs Type of Transmission	ITU's Line-of-Sight (LOS)
LOS/Transmission Range	$160\,\mathrm{m}$
Propagation Loss Model	Hybrid Buildings
PHY / MAC	LTE release 14, half-duplex
Simulation Time	$100\,\mathrm{s}$
User Service Request Model	Google Cloud HPC traces [50]

Besides PRUDENT, we have implemented three other mobility management and UAVBS deployment strategies for the same simulation scenario. Firstly, we consider the UAV positioning scheme proposed by Sun et al. [51] called STABLE, which maximizes the spectral efficiency of UAV deployments considering a half-duplex in-band mode. STABLE is a reactive scheme, which means that users are offloaded only after the congestion is detected, but not proactively. Furthermore, it considers an optimization process, which can be computationally intensive. Besides STABLE, we implemented the deployment strategy proposed by Rahman et al. [52], which proposes a UAV deployment strategy based aimed at maximizing network throughput via a heuristic to find the optimal locations for UAVBSs. However, such an approach is also reactive and may need to be recalculated as users move, potentially taking a significant amount of time in case of high user density. Finally, we implemented a baseline scenario without any UAVBS deployment. Note that in this No-UAV approach, we deploy an additional number of ground base stations corresponding to the UAVBSs present in other approaches in order to assess their performance fairly. In the No-UAV scenario, users follow the same mobility patterns and application requests as before, but only ground base stations are available to the users.

The following metrics are collected in the evaluation of the algorithms: i) The number of neural network parameters stored in the edge distributed storage; ii) The Root Mean Squared Error (RMSE) of the link usage prediction performed, in terms of the value of the predicted downlink usage per user in $\mathrm{Mbit/s}$; iii) The service level of the network, in terms of the fraction of users able to consume the minimum amount of bandwidth required by their applications, as defined in the link

[1]https://www.nsnam.org/
[2]https://sumo.dlr.de/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

TABLE III
DISTRIBUTED LEARNING OPTIMIZATION RESULTS.

	Number of Stored Parameters	RMSE (m)
PRUDENT	$9.57 \cdot 10^4$	1.11
FedAvg	$9.57 \cdot 10^3$	2.65
No-clustering	$1.55 \cdot 10^6$	0.43

usage trace in place; iv) The delay distribution for the user-generated traffic flows in the network; v) The total network throughput in $\mathrm{Mbit/s}$, measured as the average downlink throughput achieved by all users over the simulation time.

B. Experimental Results

In our experiments, we trained the neural network models for a random set of 100 users. For each user, we consider 80% of raw data for training and 20% for test. For this experiment, we consider raw user data stored at the edge servers to feed to an LSTM neural network, such as introduced in Section IV. We show the sequences in which the learning is performed as the sequence of downlink usage historical data for each user. However, learning is performed in the same manner as in the users' mobility data. In this way, the different prediction algorithms must estimate the downlink usage in the next interval based on the previous ten samples. We considered PRUDENT, FedAvg, and the no-clustering algorithms for traffic flow prediction. The FedAvg algorithm [30] is a traditional DML algorithm, which serves as a baseline. FedAvg aggregates all neural networks into a single network by averaging the trained weights of all users. The no-clustering algorithm applies a non-distributed ML algorithm to train a single neural network for each user. Finally, PRUDENT considers the proposed similarity metric to cluster users with similar statistical features, such that the features learned by the neural networks do not cancel each other when averaged into an aggregated model, as introduced in Section IV-B.

Table III shows the number of neural network parameters that must be stored in the edge servers for different traffic flow prediction algorithms and their impact on the edge servers' storage capabilities. The use of the proposed user similarity metric is used in order to cluster the neural network models of similar users. As users with similar mobility patterns yield neural networks with similar features after the training process, the same neural network can be trained for multiple users, requiring fewer networks to be stored in the edge servers, which have limited storage capabilities. Thus, by analyzing user similarity and clustering, we store a single neural network for each user cluster, which can be used to make predictions for any given user in the cluster. By analyzing the results, we conclude that clustering neural network models, as done by PRUDENT can reduce the number of neural network weights stored in the DMLS by up to 93.8% than the no-clustering algorithm. PRUDENT's storage requirements are lower because the network is required to store only a small number of neural network models (one set per cluster formed). Following this logic, the FedAvg aggregation provides the best space-reduction properties among all compared approaches because it requires storing a single neural network for all users but provides a higher RMSE.

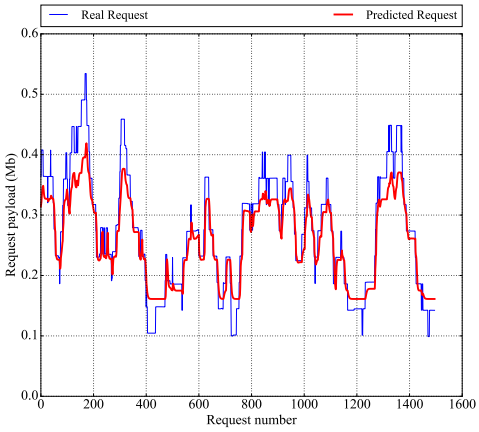


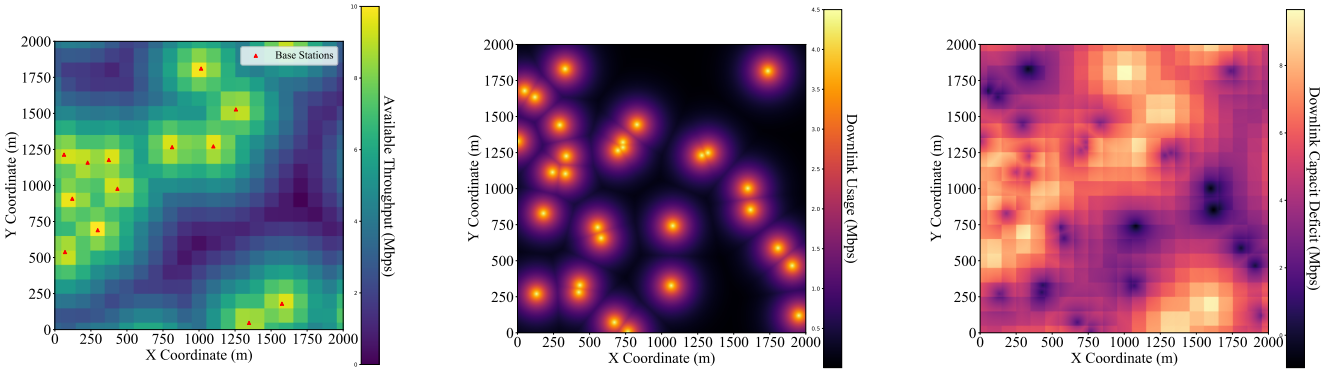
Fig. 7. Real and predicted downlink usage for a randomly selected user

Table III also shows the RMSE performance for traffic flow prediction considering different prediction algorithms. PRUDENT can deliver RMSE up to 59.2% lower than FedAvg. Furthermore, the models formed in PRUDENT have a better accuracy due to being built for users with similar statistical features. Hence, even though in PRUDENT the network must store more neural network parameters than in FedAvg, it can deliver predictions of significantly better quality, as seen in Figure III. This is because, in contrast to traditional FedAvg model aggregation, we only aggregate models from users who already have similar statistical features. While in FedAvg, certain learned features from different users may cancel each other in the aggregation step, in PRUDENT, similar features are grouped in the same models. The clustered neural network models built by PRUDENT are then used to predict actual user downlink usage in a mobile network as the usage value in a given moment in $\mathrm{Mbit/s}$. The aggregated models can predict with a satisfactory performance, as they can be applied to many users with similar statistical features.

Figure 7 shows the real and predicted downlink usage history for a single user, highlighting that the real usage is accurately predicted by the model (RMSE=$0.04\,\mathrm{Mbit/s}$). Since the prediction is produced by a clustered model, the noisier request behavior of this specific user is not closely predicted, as it is likely not a relevant behavior for the other users of the cluster. This shows that even for individual users, clustering and aggregation proposed by PRUDENT are able to provide reliable predictions for the variable relevant to the offloading and UAVBS positioning decisions.

Figure 8 shows the scenario's throughput maps to detect overloaded or void areas, as described in Section V-A. We define void areas as areas in the scenario where users cannot do access the desired bandwidth for their applications according to the trace used. Void areas depend not only on ground base station deployment but also on users' locations and requested throughput. Figure 8(b) shows real user and base station locations from the scenario, coupled with downlink usage values contained in the requests dataset. Figure 8(a) shows the spatial distribution of the throughput offered by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management



(a) Throughput offered by the ground base stations in the Köln scenario. The red dots represent the ground base stations' positions.

(b) Throughput requested by the mobile users. The red dots represent the mobile users' positions.

(c) Difference between the throughout offered by the ground base stations and the throughput requested by the mobile users.

Fig. 8. Throughput maps for the simulated scenario at the beginning of the simulation

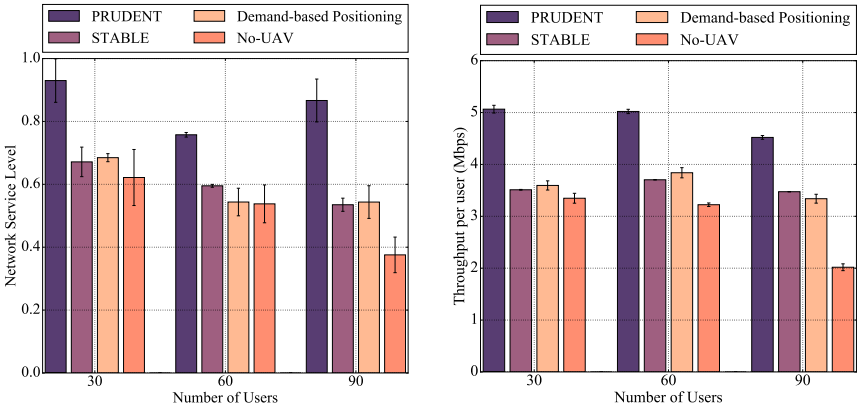
the ground base stations, as well as the estimated throughput offered by the cell at a given location, based on the expected SINR and link bandwidth in said location. This throughput configuration is calculated at the beginning of PRUDENT execution. In the considered scenario, we have 13 ground base stations deployed to cover an area of $2\,\mathrm{km} \times 2\,\mathrm{km}$ with n mobile users, considering square cells R_c having a $100\,\mathrm{m}$ edge each. As expected, regions closer to the base station location have a higher throughput (e.g., $10\,\mathrm{Mbit/s}$) compared to regions far from the base station. As we can see in Figure 8(a), areas in the scenario are not uniformly served. Instead, there are regions with significantly lower available throughput. The offered throughput map can be computed offline by the network infrastructure, as the ground base station's positions and characteristics do not change over time. Figure 8(b) shows the actual user locations, as well as the downlink usage they request. Figure 8(b) shows the throughput requested by 30 mobile users in the scenario, each requesting up to $5\,\mathrm{Mbit/s}$ of downlink traffic at their respective locations. Figure 8(c) shows the throughput deficit map used to detect overloaded or void areas. It is computed as the difference between the throughput offered by the base station and the throughput requested by the mobile users. As expected, users located distant from base stations may not be served with the minimum requirements of their applications. This configuration is realistic, as the considered base station deployment is extracted from a real-world dataset. For instance, the darker areas of the plot correspond to the overloaded regions of the scenario, in which user requests are close to or surpass the throughput offered by the ground base stations. Hence, the throughput deficit map is used by PRUDENT to determine to which base station a user should connect and where to position the UAVBSs. Since the main goal of PRUDENT is network performance optimization in terms of user request satisfaction, we evaluate the service level and throughput per user for different management strategies for offloading and UAVBS positioning in the scenario.

Figure 9 shows the network performance considering the different number of users requesting services and also different management strategies for offloading and UAVBS positioning in the scenario. We define the service level as the fraction of users able to access throughput equal or greater to the amount necessary for their applications. As the throughput requested by users varies during the simulation, we average the service level values over time for each of the algorithms tested. By analyzing the results of Figure 9(a), we can see that the service level achieved by PRUDENT is considerably superior to the other tested algorithms, as it is able to deliver a 36.7%, 27.1%, and 53.7% higher service level when compared to STABLE, the Demand-based positioning, and the No-UAV algorithms, respectively. Since PRUDENT only focuses on users who are currently served below their requests, it can further increase service level and maximize network throughput. Furthermore, the predictive approach introduced in PRUDENT can better balance the time UAVBSs take to reach predicted centroids.

Figure 9(b) depicts the average throughput per user in each of the tested scenarios, as well as for each of the algorithms tested. By analyzing the results, we can see that users receive, on average, a 31% higher throughput under PRUDENT, when compared to other algorithms. This is due to the better UAVBS deployment and mobility management strategies. We observe in the simulations that the offloading of users who do not have their requirements met significantly increases the network throughput, as these are often users who demand high link usage and may not have enough throughput available. With PRUDENT, demanding users are served with higher priority, whereas less demanding users are offloaded to neighboring cells. STABLE and the Demand-based positioning algorithm lack an efficient handover mechanism between UAVBSs and ground base stations, which can significantly reduce the efficiency of the UAV deployment.

We can see the evolution of the system under each of the algorithms considering a simulation with 60 users in which UAVs start the simulation located randomly in the scenario and users consume downlink bandwidth. Figure 10 shows the average downlink throughput over the simulated time, obtained as the summation of the downlink traffic in all base stations in the network, both ground base stations, and UAVs. We can see that the network starts at a similar throughput level for all the tested approaches. However, as users are marked as under-served by PRUDENT, since the start of the simulation,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management



(a) Network Service Level (b) Throughput per user

Fig. 9. Network results considering different number of users requesting services.

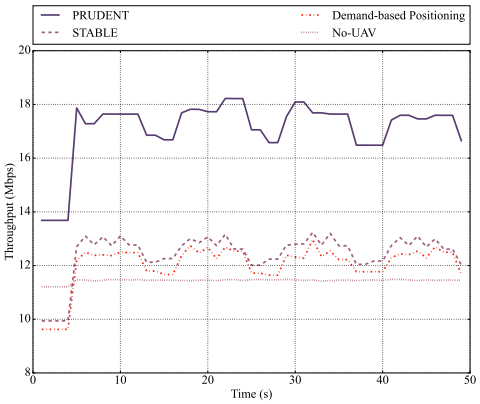


Fig. 10. Network throughput over time

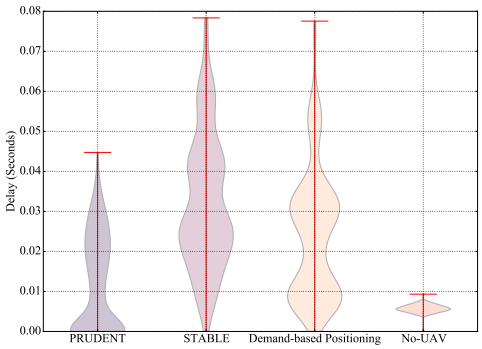


Fig. 11. Distribution of delay values for each of the algorithms tested.

requests are sent to the UAVBSs to move to the nearest under-served centroid. We can see that for PRUDENT, the first 5 seconds of the simulation have little variance in terms of network throughput. This is due to a warm-up time in which UAVBSs need to be correctly positioned and for handover requests to be made. Afterwards, we can see that network throughput significantly increases under PRUDENT. We also notice moments in which the network throughput decreases, such as near 12, 25, and 35 seconds. Such drops in network throughput can be explained by the constant user mobility, which leaves the coverage area of UAVBSs and experiences temporary lower throughput. Upon the detection of throughput drops, PRUDENT periodically evaluates the positioning of UAVBSs and connections to restore users' QoS. We can also notice that for competing approaches, such as the No-UAV, little variance over the initial throughput values is noticed, as users are allocated to ground base stations and additional base stations are not deployed through the simulation.

Figure 11 shows the distribution of delay values in the simulated traffic flows for every tested approach. We can see that PRUDENT is able to achieve shorter delays compared to the competing algorithms. The No-UAV approach achieves the most stable and lowest delay values distribution since the paths in this scenario are a single hop between user and server through the base stations. PRUDENT improves both throughput and delay compared to other UAVBS deployment approaches (STABLE and demand-based position) due to optimized network connections, meaning that users with poor connections are more likely to experience packet losses and thus increase network delay due to re-transmissions. We can also observe that the distribution of delay values under PRUDENT has a large number of very low delay measurements. While the user requests patterns are collected from real-world data traces with a specific distribution, the deployment of the UAVBSs is based on the predicted request values in order to maximize throughput for highly demanding users, thus adapting to large surges in throughput. This is because base station offloading under PRUDENT impacted the quality of the connection of individual users, either by offloading them directly or by improving the QoS in offloaded base stations. This contributes to better spectrum usage and lower transmission delays, thus reducing the overall delay for users. Delay values are measured as the end-to-end delay between user devices and the servers from which they consume applications, modeled here as UDP flows with payloads predefined by a real-world dataset [50]. It is worth noting that delays in a real-world edge-computing setup would be smaller than the simulated results, because in our considered scenario, services are provided by servers from the network core. This difference is not crucial here because, in this work we mainly focus on evaluating the impact of UAV deployment strategies on the delay at the network edge.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3123216, IEEE Transactions on Network and Service Management

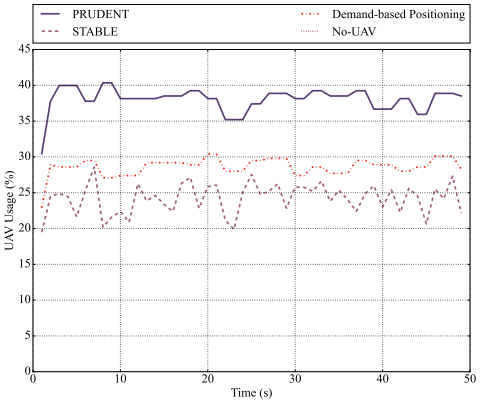


Fig. 12. Percentage of users connected to a UAVBS during the simulation.

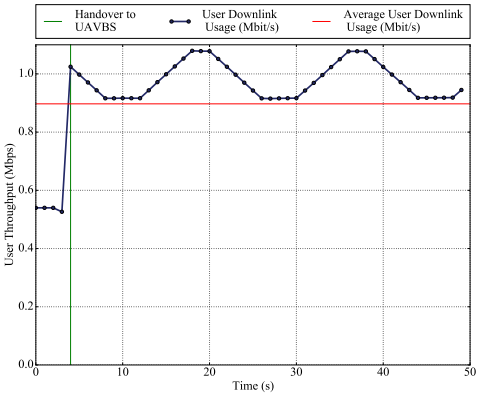


Fig. 13. Downlink network throughput before and after UAVBS offloading.

Furthermore, delay metrics are also impacted by UAVBS usage in the simulations, as we can see in Figure 12, which shows the percentage of users connected to UAVBSs during the simulated time. Note that we choose only simulations with 30 users, as the number of users may impact these values. We can see that a larger number of users is offloaded to UAVBSs under PRUDENT, given that users are offloaded due to poor QoS. This implies that the network is indeed overloaded during the simulated time. Note that in the No-UAV approach, the same number of base stations is deployed as in the other algorithms, as UAVBSs are substituted for ground base stations. We can see that the dynamism of UAVBSs allocation can greatly improve networking metrics even when compared to a similar number of deployed base stations. This is due to the fact that in a UAVBS scenario, void areas and overloaded cells can be corrected in real-time and, thus, adapt to changing usage and mobility patterns in the network.

Figure 13 shows the measured throughput over time for a single user. We can see that in the moments prior to offloading of the user to a UAVBS, the derived user throughput is provided by an overloaded cell, and their requests are served at lower throughput than the average requested. When the user is offloaded to a UAVBS, after 13 seconds from the beginning of the simulation, we observe that user requests are served within the actual requested values. The variation in values of requests is rather a feature of the user requests dataset and not a consequence of network limitations.

As we can observe, PRUDENT has the best overall performance in terms of service level and delay, which can be explained by the combination of a throughput-based UAVBS positioning with a request and mobility prediction scheme. On the other hand, the proposals STABLE and Demand-based positioning, while providing improved performance compared to the No-UAV approach in scenarios with a larger quantity of users, cannot achieve throughput and service level as good as PRUDENT due to lacking a predictive mechanism. Furthermore, PRUDENT is able to improve the accuracy of aggregated neural network parameters by using a novel similarity metric for user clustering.

VII. Conclusion

In this article, we proposed a distributed learning framework, where edge servers act as local data owners to collect connection data between mobile devices and edge servers. With the framework, we presented a proactive resource management scheme to offload user content requests and service provision to neighboring cells based on distributed predictions of user's future trajectories and content request patterns. Furthermore, we defined an UAVBS deployment strategy to serve mobile users in overloaded or shadow areas with the help of distributed group clustering solutions. Extensive simulation results showed that our framework could consistently optimize throughput and the network's service level over different scenarios. Furthermore, we improved the distributed machine learning model's accuracy via a novel similarity measure.

Acknowledge

This study was financed in part by the NSFC project with grant ID 91738301, Beihang Zhuobai program with grant ID ZG216S2176, Brazilian CNPq and CAPES project with finance code 001.

References

[1] I. Bor-Yaliniz, M. Salem, G. Senerath, and H. Yanikomeroglu, "Is 5g ready for drones: A look into contemporary and prospective wireless networks from a standardization perspective," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 18–27, 2019.

[2] M. J. Sobouti, Z. Rahimi, A. H. Mohajerzadeh, S. A. H. Seno, R. Ghanbari, J. M. Marquez-Barja, and H. Ahmadi, "Efficient deployment of small cell base stations mounted on unmanned aerial vehicles for the internet of things infrastructure," *IEEE Sensors Journal*, 2020.

[3] 3GPP, "Study on remote identification of unmanned aerial systems (uas)," 3GPP TR 22.825. Rel. 16,, Tech. Rep., 2018.

[4] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, "Uav-assisted wireless powered cooperative mobile edge computing: Joint offloading, cpu control, and trajectory optimization," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2777–2790, 2020.

[5] H. Wang, H. Ke, and W. Sun, "Unmanned-aerial-vehicle-assisted computation offloading for mobile edge computing based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 180 784–180 798, 2020.

[6] J. Lyu, Y. Zeng, and R. Zhang, "Uav-aided offloading for cellular hotspot," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 3988–4001, 2018.

[7] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2017.

[8] L. Liang, H. Ye, and G. Y. Li, "Toward intelligent vehicular networks: A machine learning framework," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 124–135, 2019.

[9] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[10] F.-L. Luo, *Machine Learning for Future Wireless Communications*. Wiley-IEEE Press, 2020.

[11] C. H. Liu, X. Ma, X. Gao, and J. Tang, "Distributed energy-efficient multi-uav navigation for long-term communication coverage by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1274–1285, 2020.

[12] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 38–67, 2020.

[13] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous internet of things: Model, applications and challenges," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1722–1760, 2020.

[14] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, "Edge intelligence: Architectures, challenges, and applications," 2020.

[15] E. Peltonen, M. Bennis, M. Capobianco, M. Debbah, A. Ding, F. Gil-Castiñeira, M. Jurmu, T. Karvonen, M. Kelanti, A. Kliks *et al.*, "6g white paper on edge intelligence," *arXiv preprint arXiv:2004.14850*, 2020.

[16] H. Wei, H. Luo, and Y. Sun, "Mobility-aware service caching in mobile edge computing for internet of things," *Sensors*, vol. 20, no. 3, p. 610, 2020.

[17] N. Samuel, T. Diskin, and A. Wiesel, "Deep mimo detection," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2017, pp. 1–5.

[18] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[19] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.

[20] U. Challita, L. Dong, and W. Saad, "Proactive resource management for lte in unlicensed spectrum: A deep learning perspective," *IEEE Transactions on Wireless Communications*, vol. 17, no. 7, pp. 4674–4689, 2018.

[21] A. Jain, E. Lopez-Aguilera, and I. Demirkol, "Are mobility management solutions ready for 5G and beyond?" *Computer Communications*, vol. 161, no. November 2019, pp. 50–75, 2020.

[22] E. Montero, D. Rosario, and A. Santos, "Clustering Users for the Deployment of UAV as Base Station to Improve the Quality of the Data," *Proceedings - 2019 IEEE Latin-American Conference on Communications, LATINCOM 2019*, 2019.

[23] Z. Zhao, M. Karimzadeh, L. Pacheco, H. Santos, D. Rosário, T. Braun, and E. Cerqueira, "Mobility management with transferable reinforcement learning trajectory prediction," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2102–2116, 2020.

[24] J. Ding, H. Liu, L. T. Yang, T. Yao, and W. Zuo, "Multiuser multivariate multiorder markov-based multimodal user mobility pattern prediction," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4519–4531, 2020.

[25] X. Feng, X. Ling, H. Zheng, Z. Chen, and Y. Xu, "Adaptive multi-kernel svm with spatial–temporal correlation for short-term traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2001–2013, 2019.

[26] S. Wang, H. Miao, J. Li, and J. Cao, "Spatio-temporal knowledge transfer for urban crowd flow prediction via deep attentive adaptation networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2021.

[27] D. H. L. Oliveira, T. P. de Araujo, and R. L. Gomes, "An adaptive forecasting model for slice allocation in softwarized networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[28] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, Oct 2014.

[29] M. Mukherjee, V. Kumar, A. Lat, M. Guo, R. Matam, and Y. Lv, "Distributed deep learning-based task offloading for uav-enabled mobile edge computing," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 1208–1212.

[30] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *arXiv preprint arXiv:1602.05629*, 2016.

[31] S. Hu, X. Chen, W. Ni, E. Hossain, and X. Wang, "Distributed machine learning for wireless communication networks: Techniques, architectures, and applications," 2020.

[32] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[33] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[34] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," *arXiv preprint arXiv:1712.07628*, 2017.

[35] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 10, pp. 4229–4238, 2019.

[36] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, 2019.

[37] T. T. Vu, D. T. Ngo, N. H. Tran, H. Q. Ngo, M. N. Dao, and R. H. Middleton, "Cell-free massive mimo for wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6377–6392, 2020.

[38] O. Habachi, M.-A. Adjif, and J.-P. Cances, "Fast uplink grant for noma: a federated learning based approach," 2019.

[39] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, "3-d placement of an unmanned aerial vehicle base station (uav-bs) for energy-efficient maximal coverage," *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 434–437, 2017.

[40] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-uav assisted wireless networks: A machine learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7957–7969, 2019.

[41] C. Lai, C. Chen, and L. Wang, "On-demand density-aware uav base station 3d placement for arbitrarily distributed users with guaranteed data rates," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 913–916, 2019.

[42] A. V. Savkin and H. Huang, "Deployment of unmanned aerial vehicle base stations for optimal quality of coverage," *IEEE Wireless Communications Letters*, vol. 8, no. 1, pp. 321–324, 2019.

[43] P. Yang, X. Cao, C. Yin, Z. Xiao, X. Xi, and D. Wu, "Proactive drone-cell deployment: Overload relief for a cellular network under flash crowd traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2877–2892, 2017.

[44] L. P. Qian, Y. Wu, H. Zhou, and X. Shen, "Dynamic cell association for non-orthogonal multiple-access v2s networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 10, pp. 2342–2356, 2017.

[45] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197 017–197 046, 2020.

[46] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings 18th international conference on data engineering*. IEEE, 2002, pp. 673–684.

[47] R. Arshad, H. Elsawy, S. Sorour, T. Y. Al-Naffouri, and M. S. Alouini, "Cooperative handover management in dense cellular networks," *2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings*, 2016.

[48] E. Demarchou, C. Psomas, and I. Krikidis, "Mobility Management in Ultra-Dense Networks: Handover Skipping Techniques," *IEEE Access*, vol. 6, pp. 11 921–11 930, 2018.

[49] S. Uppoor and M. Fiore, "Large-scale urban vehicular mobility for networking research," in *2011 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2011, pp. 62–69.

[50] A. Hussain and M. Aleem, "GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures," *Data*, vol. 3, no. 4, pp. 1–12, 2018.

[51] X. Sun and N. Ansari, "Jointly optimizing drone-mounted base station placement and user association in heterogeneous networks," *IEEE International Conference on Communications*, vol. 2018-May, pp. 0–5, 2018.

[52] S. Rahman and Y. Z. Cho, "UAV positioning for throughput maximization," *Eurasip Journal on Wireless Communications and Networking*, vol. 2018, no. 1, 2018.